PASSED

732A38: Computer lab 3
Computational statistics

Martina Sandberg and Caroline Svahn

March 3, 2016

1 Cluster sampling

An opinion pool is assumed to be performed in several locations of Sweden by sending interviewers to this location. Of course, it is unreasonable from the financial point of view to visit each city. Instead, a decision was done to use random sampling without replacement with the probabilities proportional to the number of inhabitants of the city to select 20 cities. Explore the file *population.xls*. Note that names in bold are counties, not cities.

1.1

Import necessary information to R.

1.2

Use uniform random number generator to create a function that selects 1 city from the whole list by the probability scheme offered above (do not use standard sampling functions present in R).

See 3), which is a updated version of this function.

1.3

Use the function you have created in step 2 as follows:

1. Apply it to the list of all cities and select one city
2. Remove this city from the list
3. Apply this function again to the updated list of the cities
4. Remove this city from the list
5. ... and so on until you get exactly 20 cities.

First, the probabilities of the cities are calculated as

$$p = \frac{Population_{city}}{Population_{Total}}$$

Then one observation is drawn from the uniform distribution and compared to cumulative intervals in a loop. If the value of this observation (U) is greater than the sum of the cumulative probability of the current index, the current city is stored and evaluated again, for a higher cumulative probability. When U is smaller than the cumulative probability, the current city is chosen and the loop is stopped with a break. The city is then deleted from the dataset and the whole procedure is repeated until the number of k (requested draws) is met.

1.4

Run the program. Which cities were selected? What can you say about the size of the selected cities?

The obtained cities are:

City	Population
Skövde	50984
Älvdalen	7288
Arvika	26100
Gävle	94352
Helsingborg	128359
Stockholm	829417
Växjö	82023
Åstorp	14667
Ulricehamn	22753
Luleå	73950
Huddinge	95798
Uppsala	194751
Åckerö	12292
Botkyrka	81195
Kristianstad	78788
Skurup	14867
Klippan	16382
Lund	109147
Nyköping	51209
Skellefteå	71770

Table 1: The 20 selected cities from the implemented function.

where

Cities	Min	Max	Mean	Median
20	7288	829417	102804.6	72860
All	2500	829417	32209.25	15282

Table 2: Values for the population of the 20 selected cities and all cities.

In table 1 the selected cities from the function is shown. In table 2 some population values can be seen from all cities and from just the 20 selected cities. Here it is clear that the selected cities from the function are among the bigger ones. This is reasonable since the larger the city, the wider the probability interval and therefore, the higher the probability to be drawn.

1.5

Plot one histogram showing the size of all cities of the country. Plot another histogram showing the size of the 20 selected cities. Conclusions?

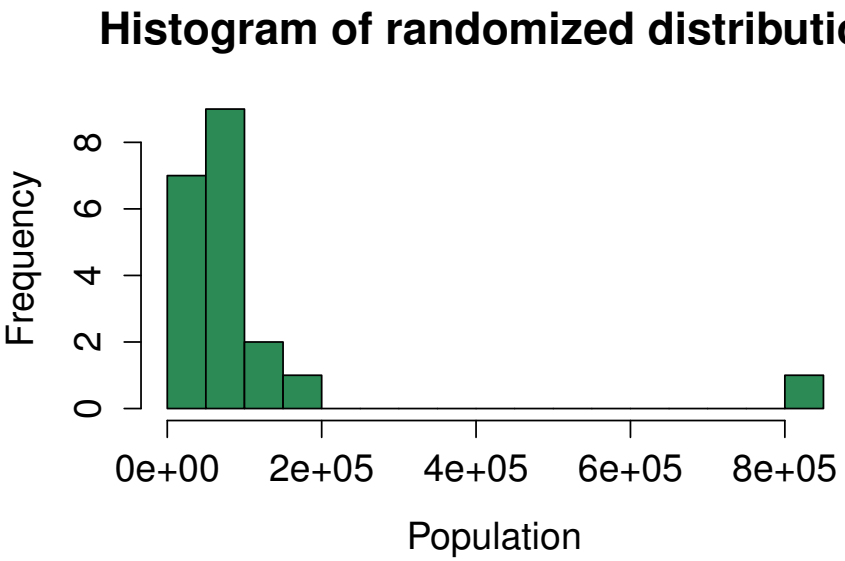

Figure 1: Distribution of number of inhabitants in the selected 20 cities.

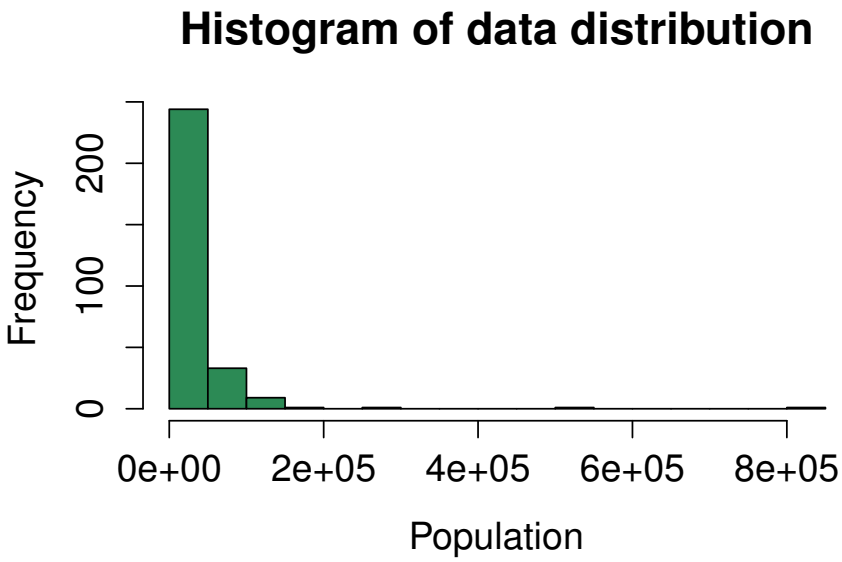

Figure 2: Distribution of number of inhabitants in all cities.

In figure 1 the distribution of number of inhabitants in the selected 20 cities are shown in a histogram. Likewise the distribution of number of inhabitants in all cities are shown in figure 2. From before we noticed that the selected cities are a sample represented by most bigger cities. This is also clear in these histograms where we can conclude that the selected 20 cities are overrepresented by big cities.

Regarding the sampling method - depending on what the goal is, this approach may be unsuitable. If the aim is to draw 20 cities representing Sweden as a country when investigating some other variable to make conclusions about the entire country, the results are probably misleading. This since most of the larger cities are located in the south of Sweden - Stockholm, Uppsala, Gothenburg, Linköping, Norrköping, Jönköping etc, are all located rather close to each other. Meanwhile, in the north, the density of the inhabitants is substantially smaller but the area is still very wide. Thus, the sample drawn is likely to represent the south of Sweden. To avoid this skewness, allocating the sample by area, for example, is adviced.

2 Different distributions

The double exponential (Laplace) distribution is given by the formula:

$$DE(\mu, \alpha) = \frac{\alpha}{2} exp\{-\alpha|x - \mu|\}$$

2.1

Write a code generating double exponential distribution DE(0,1) from U(0,1) by using inverse CDF method. Explain how you obtained that code step by step. Generate 10000 random numbers from this distribution, plot the histogram and comment whether the result looks reasonable.

$$f_X(x|\mu = 0, \alpha = 1) = \frac{1}{2}e^{-|x|} \tag{1}$$

$$f_X(x) = \begin{cases} \frac{1}{2}e^{x}, & x < 0 \\ \frac{1}{2}e^{-x}, & x \geq 0 \end{cases} \tag{2}$$

$$F_X(x) = \int_{-\infty}^{x} \frac{1}{2}e^{u}du = \frac{1}{2}\left[e^{u}\right]_{-\infty}^{x} = \frac{1}{2}e^{x} \tag{3}$$

$$\left.\begin{aligned} F_X(x) = \int_{-\infty}^{x} \frac{1}{2}e^{-u}du &= \int_{-\infty}^{0} \frac{1}{2}e^{-u}du + \int_{0}^{x} \frac{1}{2}e^{-u}du \\ = \frac{1}{2}e^{-0} + \frac{1}{2}\left[-e^{-u}\right]_{0}^{x} &= \frac{1}{2} - \frac{1}{2}\left(e^{-x} - e^{-0}\right) \\ &= \frac{1}{2} - \frac{1}{2}e^{-x} + \frac{1}{2} = 1 - \frac{1}{2}e^{-x} \end{aligned}\right\} \tag{4}$$

$$F_X(x) = \begin{cases} \frac{1}{2}e^{x}, & x < 0 \\ 1 - \frac{1}{2}e^{-x}, & x \geq 0 \end{cases} \tag{5}$$

$$y = \frac{1}{2}e^{x} \Rightarrow 2y = e^{x} \Rightarrow x = ln(2y) \tag{6}$$

$$y = 1 - \frac{1}{2}e^{-x} \Rightarrow 2 - 2y = e^{-x} \Rightarrow x = -ln(2 - 2y) \tag{7}$$

$$F_X(y) = \begin{cases} ln(2y), & y\epsilon(0, \frac{1}{2}) \\ -ln(2 - 2y), & y\epsilon[\frac{1}{2}, 1) \end{cases} \tag{8}$$

The double exponential distribution DE(0,1) can be written as (1). A realization of the random variable X with CDF F_X can now be obtained by $X = F_X^{-1}(U)$.

When deriving the CDF we divide into two cases (2). We have now for $x < 0$ (3) and for $x \geq 0$ (4). Thus we have that $F_X(x)$ can be expressed as (5). To find F_X^{-1} we solve the equations $y = e^x/2$ and $y = 1 - e^x/2$ for x. These computations can be seen in (6) and (7). Thus we have that $F_X(y)$ is expressed as (8). In our function we now generate a random number from U(0,1), if it is less that 1/2 we use the first formula to generate the number from the Laplace distribution and if it is grater or equal to 1/2 we use the second one. We do this 10000 times to get 10000 random numbers from the Laplace distribution.

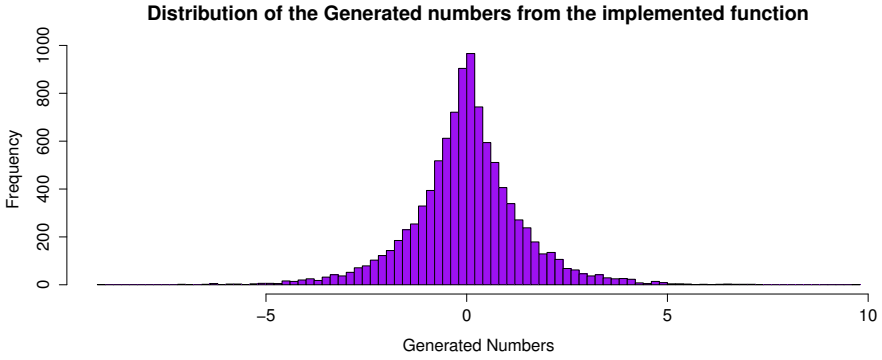

Figure 3: Distribution of the 10000 random numbers generated from the implemented function.

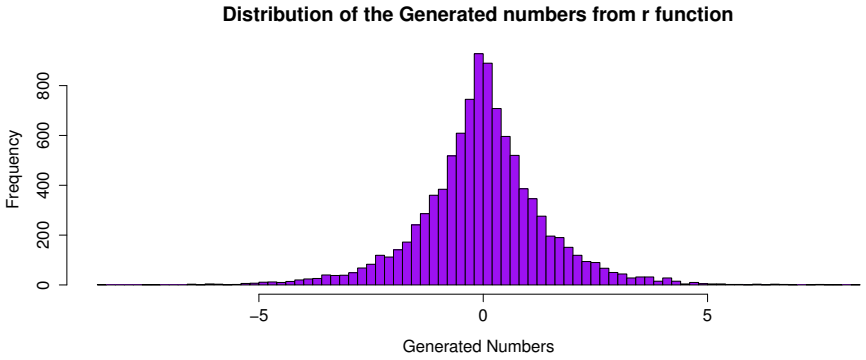

Figure 4: Distribution of 10000 random numbers generated from a Laplace-function inside `R`.

In figure 3 the 10000 random numbers generated from our function is plotted. In figure 4 10000 random numbers generated from a Laplace-function inside `R` is plotted. The histograms are very similar - therefore, our results does appear reasonable.

2.2

Use Acceptance/rejection method with DE(0,1) as majorizing density to generate N(0,1) variables. Explain step by step how this was done. How did you choose constant c in this method? Generate 2000 random numbers N(0,1) using your code and plot the histogram. Compute the average rejection rate R in the acceptance/rejection procedure. What is the expected rejection rate ER and how close is it to R? Generate 2000 numbers from N(0,1) using standard rnorm procedure, plot the histogram and compare the obtained two histograms.

First we look at the formula $cf_Y(x) \geq f_X(x)$ where $f_Y(x)$ is a majorizing density (or proposal density), $f_X(x)$ is the target density and c is a majorizing constant. We want to find the optimal value for c, which is the samallest one possible. We do this by first computing all $f_X(x)/f_Y(x)$ and then choose the biggest of these values, because this is the smallest value of c that can be picked that covers the target density function. The optimal c shown to be 1.315. Now we use the acceptance/rejection algorithm, which generate Y from the DE(0,1) distribution and U from U(0,1). Now if $U \leq f_X(Y)/cf_Y(Y)$, Y is accepted and otherwise it is rejected. We do this until we get 2000 accepted Y-values. A histogram of these Y-values can be seen in figure 5. The expected acceptance rate is $1/c$ and thus the expected rejection rate (R) is computed as $E(R) = 1 - 1/c = 0.240$. When generating the 2000 Y-values 645 was rejected. The average rejection rate in this acceptance/rejection procedure is therefore $1 - 2000/2645 = 0.244$. Our rejection rate is quite close to the expected R. A histogram of 2000 numbers from N(0,1) using standard rnorm procedure in `R` can be seen in figure 6. The two obtained histograms are very similar so we can conclude that we have in fact generated N(0,1) variables.

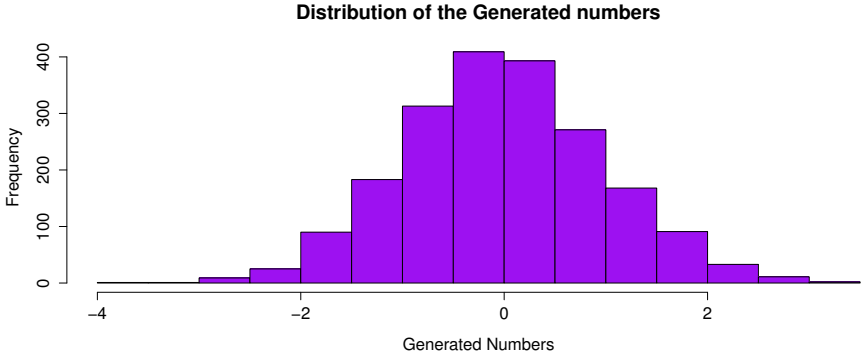

Figure 5: Distribution of the 2000 numbers generated from the implemented function.

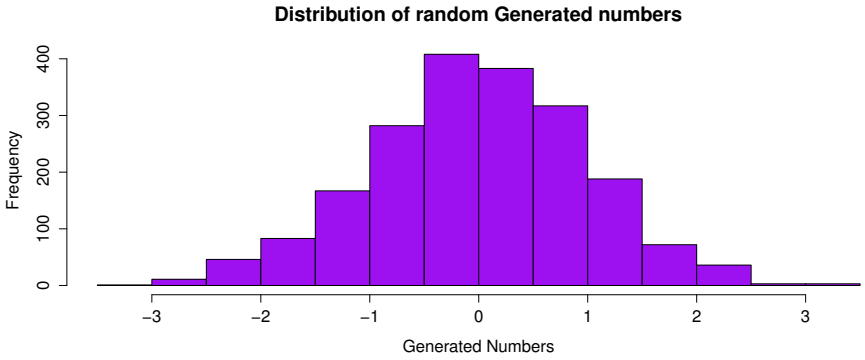

Figure 6: Distribution of 2000 numbers generated from the standard rnorm function in R.

3 Contributions

The report is rather nested by the individual reports - some plots are from Caroline's report and some from Martina's. Chunks of text have been taken from each report and then merged. The code is, this time, from Caroline's report but since most of the code is rather trivial, both individual programs are basically the same.

Appendix

```
#1.1

library(xlsx)
pop<-read.xlsx("D:/Dokument/732A38/Lab3/population.xls", sheetIndex=2)

#1.2
n<-dim(pop)[1]
sumPop<-sum(pop$Population)

for (i in 1:n) {
  pop$Prob[i]<-pop$Population[i]/sumPop
}

#1.3
rand<-function(a,x,c,m,k,pop) {
  xs<-numeric()
  xs[1]<-x
  cities<-NULL
  probab<-NULL
```

```
  for (j in 2:(k+1)) {

    n<-dim(pop)[1]

    #Probabilities
    for (i in 1:n) {
      pop$Prob[i]<-pop$Population[i]/sum(pop$Population)
    }

    #Generate U
    #xs[j]<-(a*xs[j-1]+c)%%m
    #U<-xs[j]/m
    U<-runif(1)

    for (l in 1:n) {
      temp<-pop$City[l]

      if (U>sum(pop$Prob[1:l])) {

        temp<-temp
      } else {
        probab<-append(probab,pop$Population[l])
        pop<-pop[-l,]
        cities<-append(cities,as.character(temp))
        break
      }
    }
  }
  return(list(cities,probab))
}

set.seed(12345)
f<-rand(a=2,x=3,c=3,m=200,k=20,pop=pop)

#1.5
hist(pop$Population, breaks=20, xlab="Population", col="seagreen",
        main="Histogram of data distribution")
hist(f[[2]], breaks=20, xlab="Population", col="seagreen",
        main="Histogram of randomized distribution")

#2.1
inv<-numeric()
for (i in 1:10000) {
  U<-runif(1)

  if (U<0.5) {
    Xt<-log(2*U)
  } else {
    Xt<-(-log(2-2*U))
  }
```

```
  inv<-append(inv,Xt)
}

hist(inv, breaks=80, xlab="Generated", col="seagreen",
        main="Histogram of Laplace distribution")

#2.2

install.packages("smoothmest")
library(smoothmest)

Y<-numeric()
cv<-numeric()
Yv<-numeric()
rejected<-0

for (i in -10:10) {
  cv<-append(c,dnorm(i)/ddoublex(i,mu=0,lambda=1))
}

c<-max(cv)

while (length(Yv)<2000) {

  U<-runif(1)
  Y<-rdoublex(1,mu=0,lambda=1)
  N<-dnorm(Y)
  X<-ddoublex(Y,mu=0,lambda=1)

  if (U <= N/(c*X)) {
    Yv<-append(Yv,Y)
  } else {
    rejected<-rejected+1
  }
}

hist(Yv, col="seagreen")
hist(rnorm(2000), col="seagreen")

rRate<-rejected/(rejected+2000)
expRate<-1-(1/c)
```